AFRL-IF-RS-TR-2002-288
Final Technical Report
October 2002

AUTOMATIC RESPONSE TO INTRUSION

Boeing Phantom Works

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-2002-288 has been reviewed and is approved for publication.

APPROVED:

LUIGI SPAGNUOLO
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)

2. REPORT DATE
October 2002

3. REPORT TYPE AND DATES COVERED
Final Mar 97 – Aug 02

4. TITLE AND SUBTITLE
AUTOMATIC RESPONSE TO INTRUSION

5. FUNDING NUMBERS
C - F30602-97-C-0309
PE - 62301E
PR - F025
TA - 11
WU - 21

6. AUTHOR(S)
Dan Schnackenberg, Harley Holliday, Travis Reid, Kelly Bunn, Dan Sterne, Kelly Djahandari, and Brett Wilson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Boeing Phantom Works
PO Box 3999
Seattle Washington 98124-2499

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFG
525 Brooks Road
Rome NY 13441-4504

Air Force Research Laboratory/SNHS
31 Grenier Street
Hanscom AFB MA 01731-2909

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2002-288

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Luigi Spagnuolo/SNHS/(781) 377-4249/ Luigi.Spagnuolo@hanscom.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

This contract final technical report documents the Automated Response to Intrusion project results. This project extended concepts developed in the Dynamic Cooperating Boundary Controllers project, which developed the initial version of the Intruder Detection and Isolation Protocol (IDIP). IDIP provides an infrastructure for intruder tracking and containment. The focus of the extensions developed under the Automated Response to Intrusions project was to integrate the IDIP technology with selected security technologies to improve the effectiveness of the intrusion response system. This work also leveraged results from the Adaptive System Security Policies contract. This report provides an overview of the current IDIP implementation and the results of this contract. It also provides references to IDIP documentation and technical papers where more detail can be found on the IDIP implementation and architecture.

14. SUBJECT TERMS
Intruder Detection Isolation Protocol, Common Intrusion Detection Framework, Network Intrusion Detection, Computer Security, Information Warfare

15. NUMBER OF PAGES
39

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Automated Response to Intrusions

1. INTRODUCTION

The initial objective of this effort was to integrate technology being developed for security devices to cooperatively respond automatically to network intrusion across small to very large-scale networks of networks that span numerous administrative domains. The resulting integrated intrusion response system was to be demonstrated in a series of integrated feasibility demonstrations (IFD) showing increasing capabilities to respond to red team attack. Through the course of the program, the objectives changed significantly. The objective became to support a series of experiments that attempted to improve understanding of various aspects of defense in depth. Integrated feasibility demonstrations were replaced by major experiments requiring integration of various technologies. As a result, our focus changed from developing a more sophisticated intrusion response system to supporting experiments defined by the Information Assurance program. These experiments still required integration of intrusion detection and response components, as well as development of new capabilities. However, the developed system was no longer the focus of the effort.

This project used the Intruder Detection and Isolation Protocol (IDIP), developed initially under the Dynamic Cooperating Boundary Controllers contract, as the basis for integrating security technologies into an intrusion detection and response system. The flexibility of the IDIP software proved useful for adding management functions for IDIP-enabled components. The IDIP software base provides an infrastructure for supporting real-time tracking and containment of attacks both within a host and across network boundaries. This infrastructure includes the protocols necessary for cooperation between varied intrusion detection and response components to enable attack containment.

Section 2 contains a brief overview of IDIP. Details of the IDIP architecture, application protocol, and message protocol are contained in [1], [2], and [3], respectively. Section 0 describes the accomplishments of this project, including contributions of this project to the Information Assurance program experimentation effort. Section 4 provides a conclusion and summary.

2. IDIP OVERVIEW

The IDIP concept of operations has each response component independently deciding on what is an appropriate response. The system's objective is to generate the response as close as possible to the attacker, minimizing the response impact on the critical functions of the system under attack. Each component's objective is to allow this optimal response while protecting local resources as well. This report discusses improvements to how IDIP nodes respond through better cooperation and a more consistent set of response mechanisms.

The primary notion that has evolved through this effort is that IDIP responses should be taken in two stages: (1) an initial immediate response that may be relatively harsh (i.e., may cause damage to normal system functionality), but is relatively short-lived, and (2) a more reasoned "optimal" response that is more effective at meeting the system's overall operational needs while attempting to contain the attack.

The Dynamic, Cooperating Boundary Controllers contract developed IDIP and the basic IDIP concept of operations that enables intruder tracking and containment. However, the response mechanisms developed were generally harsh (e.g., blocking incoming TELNET connections until manual intervention), potentially causing more damage to the system than the original attack. However, the infrastructure developed supported later development of better response mechanisms, and with some extensions to the protocol and response components, enabled optimal responses that can adapt to changes in the network threat environment.

Figure 2-1 shows the various components that can participate in an IDIP-based response. Intrusion detection components initiate IDIP response messages. Boundary controllers (e.g., routers and firewalls) provide network-based response mechanisms by blocking the intruder's access to network resources. Hosts provide finer-grained responses by killing processes and connections associated with intruders or disabling compromised user accounts. A centralized network management component (called the Discovery Coordinator) receives intrusion reports and audit data from other IDIP nodes enabling it to (1) provide administrative personnel with a global picture of the system intrusion status, and (2) coordinate the overall system response to attacks.

The approach in this project was to use the same underlying requirements developed during the Dynamic, Cooperating Boundary Controllers project, including providing support for responses across heterogeneous networks of networks that may span administrative boundaries. This implies that the mechanisms developed cannot be tightly integrated between different components. For example, no component knows enough about the system to determine the precise responses required. The Discovery Coordinator potentially has the knowledge required for optimal responses within its administrative domain, but knows little about remote administrative domains. The synergy gained from the Adaptive System Security Policies contract enabled development and demonstration of more capabilities than originally planned.

The overall approach was to (1) implement extensions to IDIP in response to requirements for Information Assurance program demonstrations and experiments, and (2) integrate IDIP with components as needed to achieve the functionality required for Information Assurance program demonstrations and experiments.

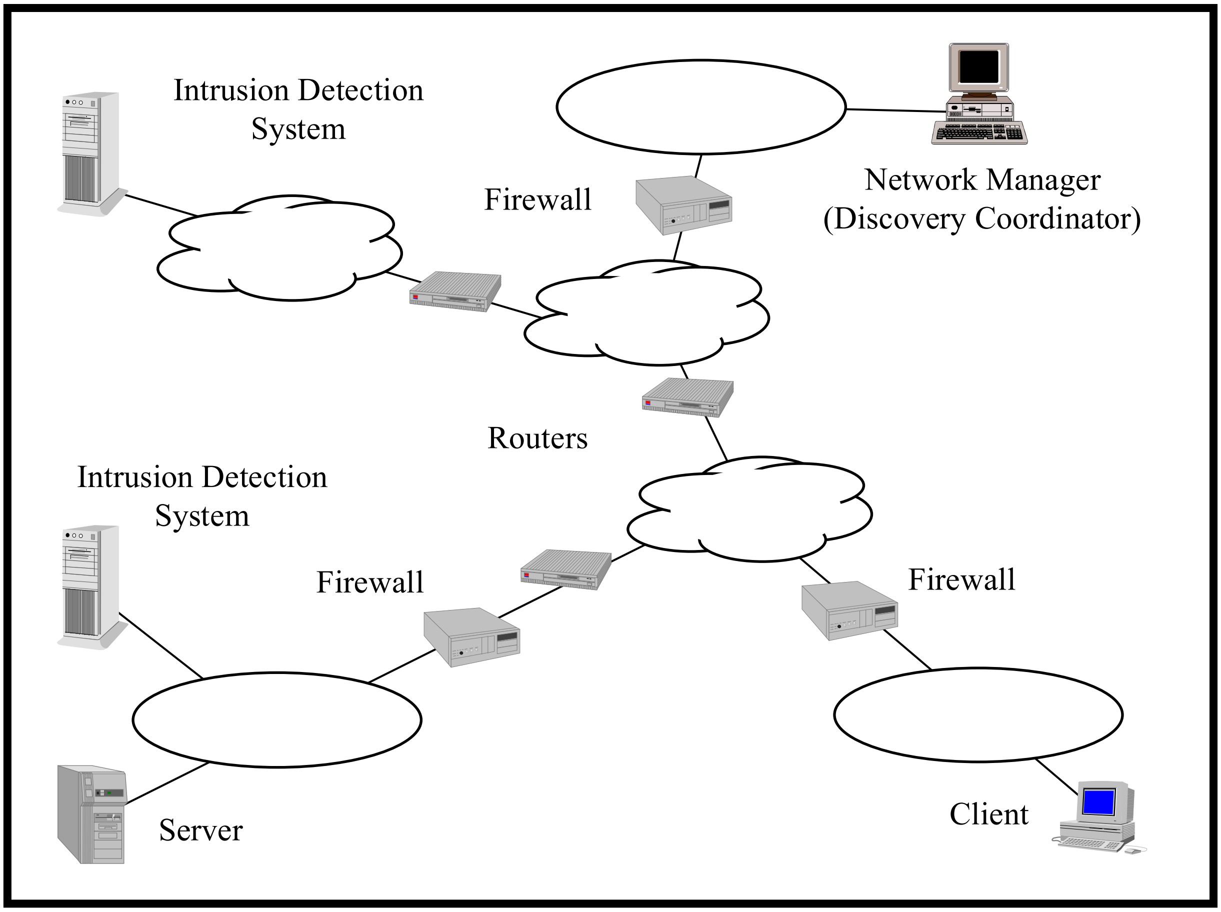

Figure 2-1. IDIP Nodes

The IDIP application layer protocol coordinates intrusion tracking and isolation. IDIP systems are organized into IDIP *communities* (as shown in Figure 2-2). Each IDIP community is an administrative domain, with intrusion detection and response functions managed by a component called the Discovery Coordinator. Communities are further organized into IDIP *neighborhoods.* These neighborhoods are the collection of components with no other IDIP node between them. Boundary control devices are members of multiple IDIP neighborhoods.

IDIP's objective is to share the information necessary to enable intrusion tracking and containment. Figure 2-3 through Figure 2-8 illustrate how IDIP accomplishes intrusion response. When an attack traverses an IDIP-protected network, each IDIP node along the path is responsible for auditing the connection or datagram stream[1].

---

[1] For brevity we will henceforth use the term *connection* generically to refer to both TCP connections and datagram packet streams.

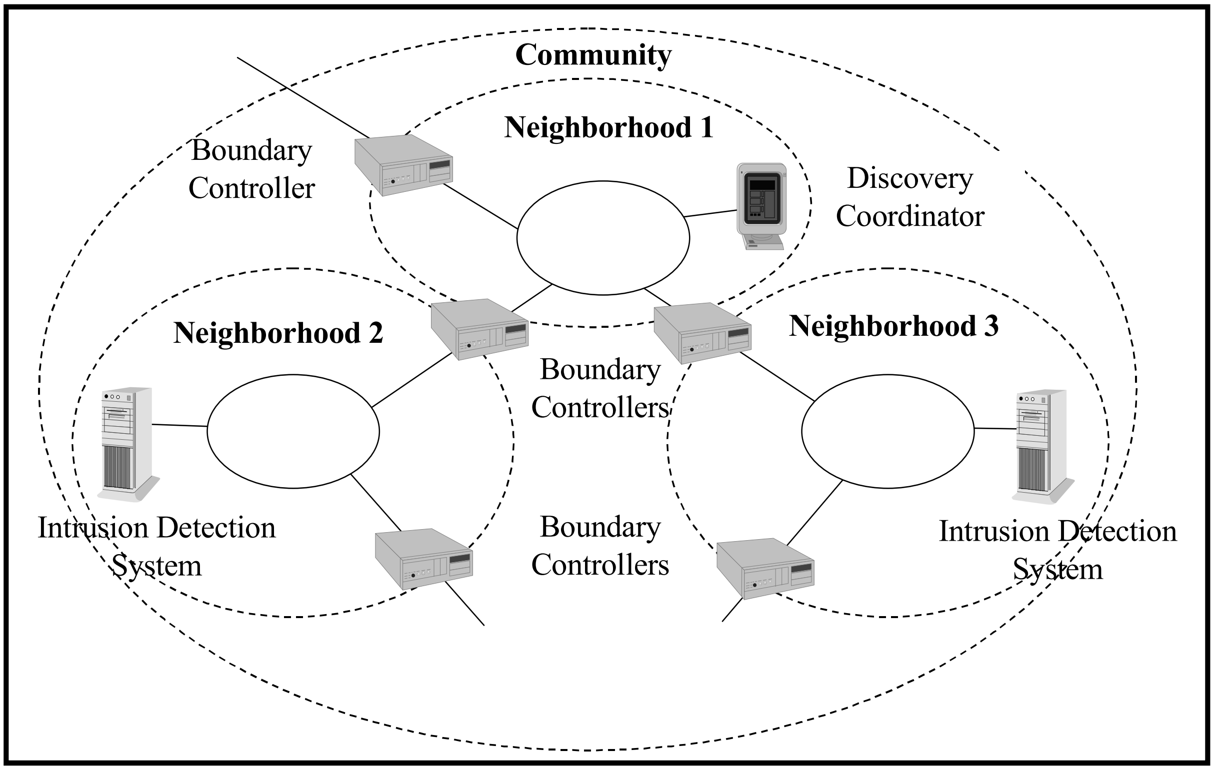

Figure 2-2. IDIP Communities

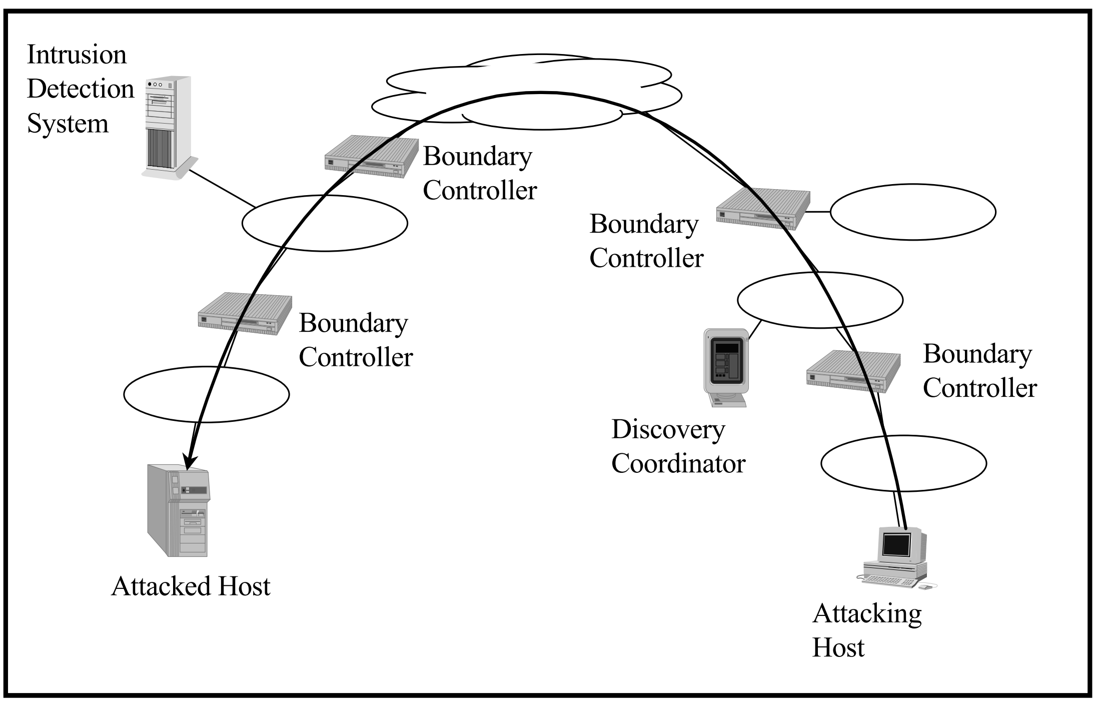

Figure 2-3. Attack Scenario

On detection of an attack, the detecting IDIP node determines the appropriate response, and if a response is indicated, notifies its neighbors of the attack. Each IDIP node makes a local decision as to what type of response (e.g., kill the connection, install filtering rules, disable the user account) is appropriate based on the attack type, attack certainty, attack severity relative to the type of attack and vulnerability of components under attack, what other IDIP nodes have already done, and local policy constraints (e.g., never disable HTTP between 8 AM and 4 PM). The types of issues addressed by the response policy are shown in Figure 2-4. The attack responses are appended to the attack description prior to forwarding the attack description to neighboring IDIP nodes. This enables IDIP to trace the attack back to the edge of the IDIP-protected system, taking appropriate responses at each IDIP node along the attack path. Nodes that receive reports from neighbors determine if they are on the attack path (i.e., whether they have seen the connection described by the attack report) before forwarding the attack report. This process continues (as shown in Figure 2-5 and Figure 2-6) until the IDIP system edge is reached.

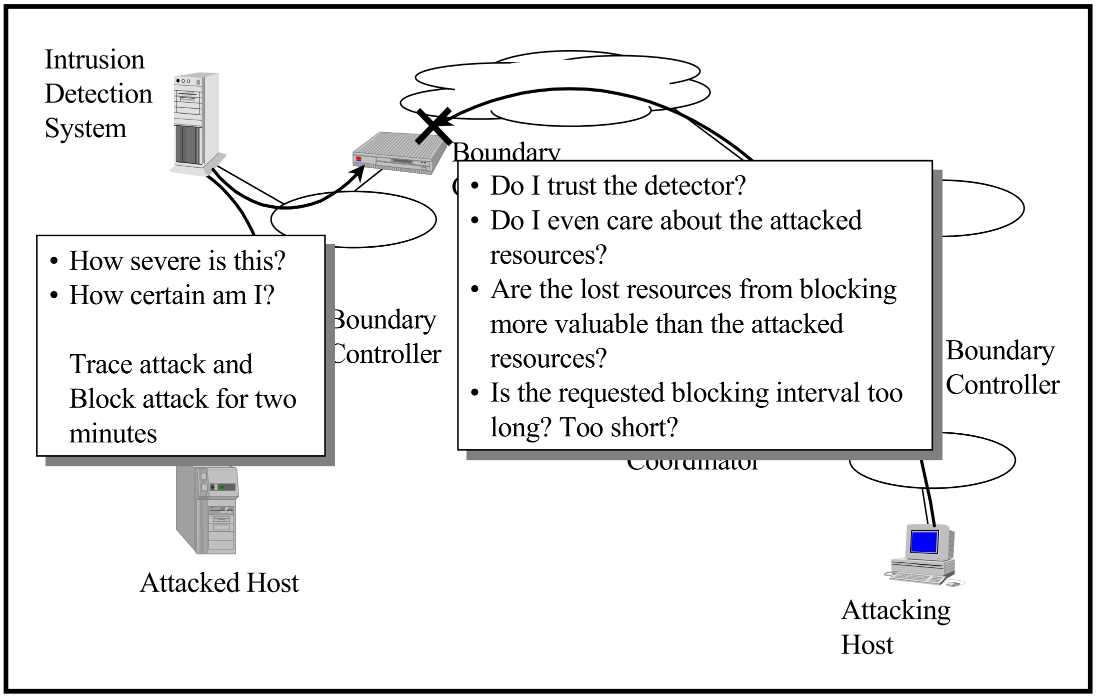

Figure 2-4. IDIP Local Neighborhood Response

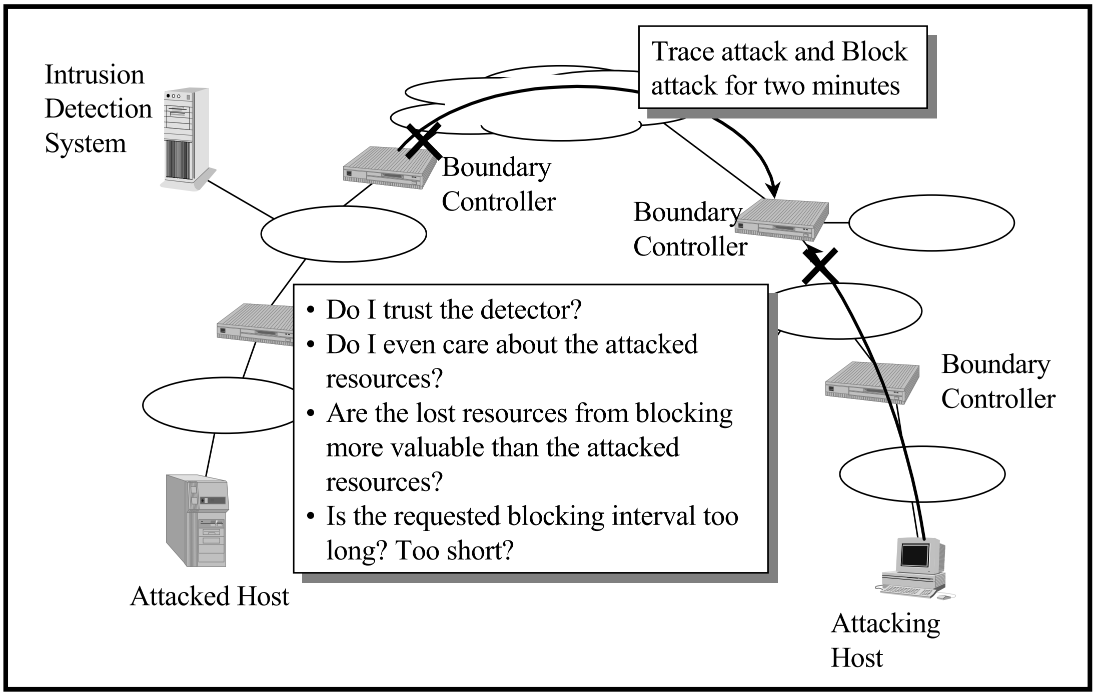

Figure 2-5. IDIP Remote Boundary Controller Response

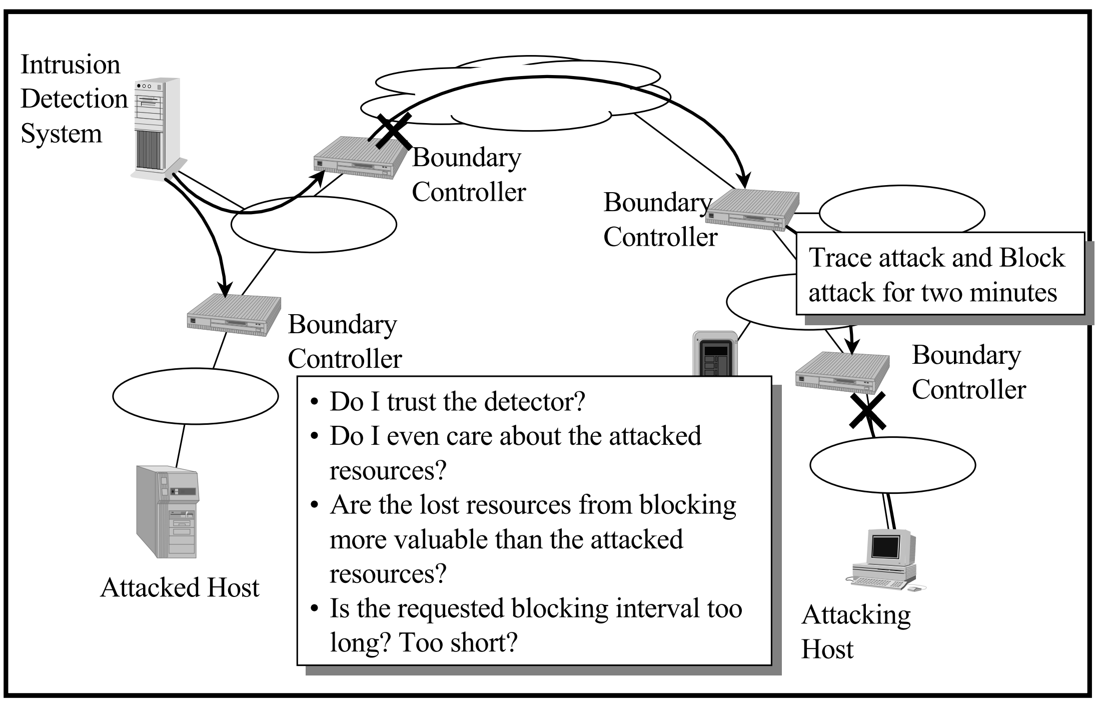

Figure 2-6. IDIP Remote Boundary Controller Response (Continued)

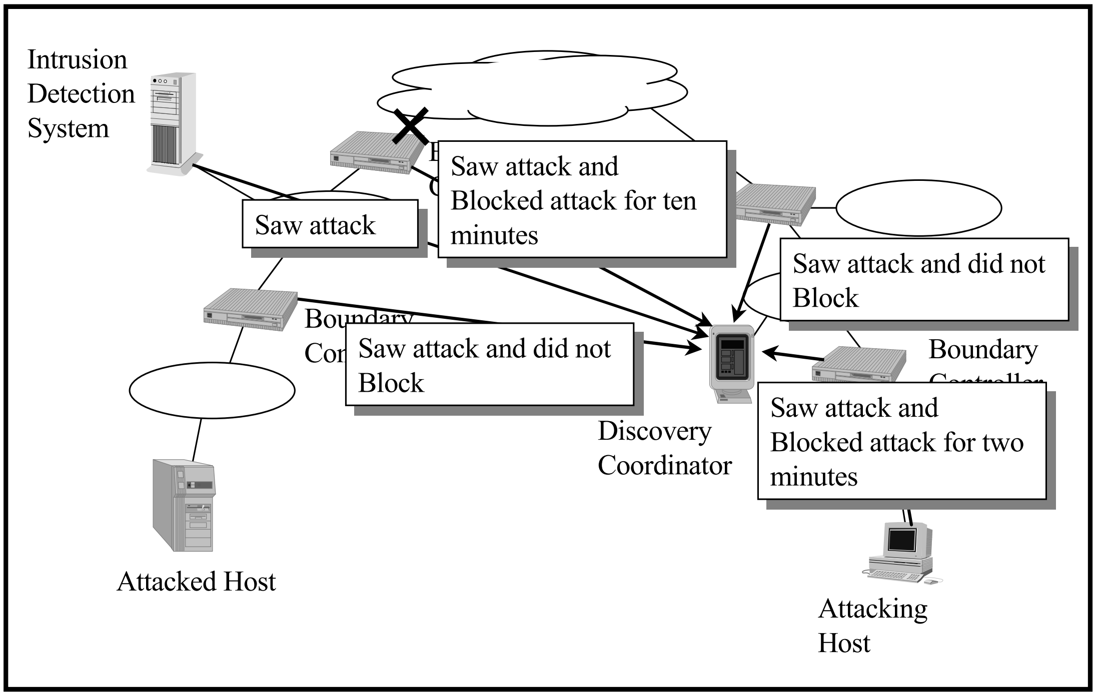

Figure 2-7. IDIP Intrusion Reporting

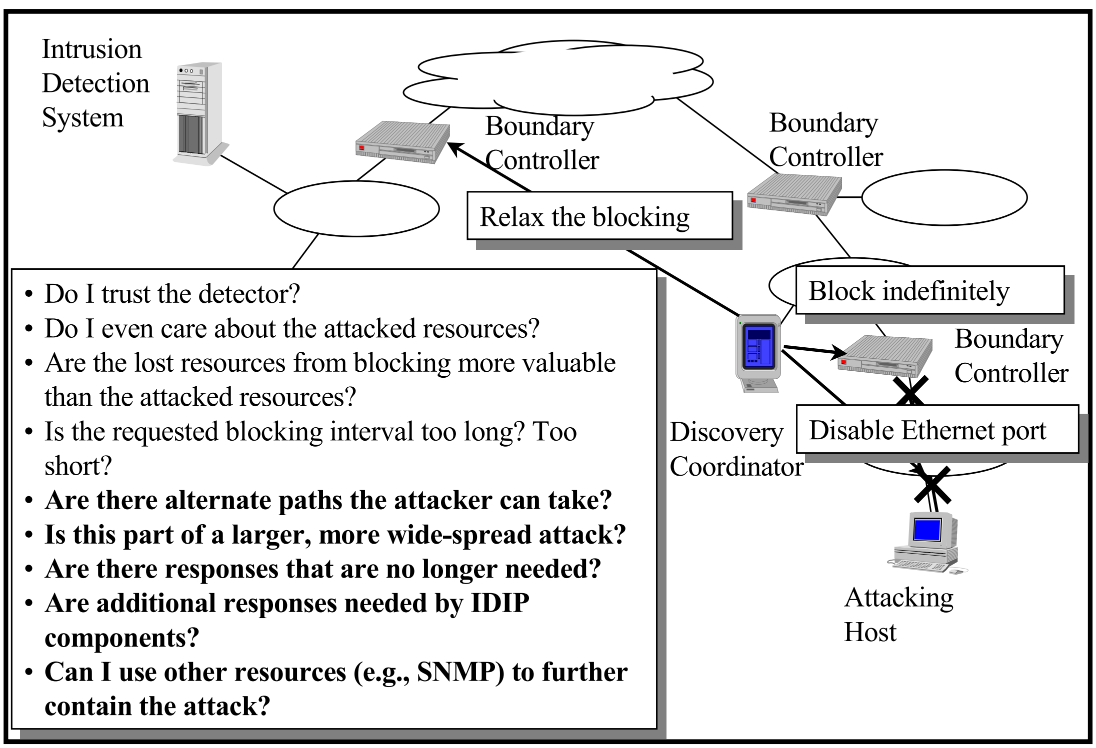

Figure 2-8. IDIP Discovery Coordinator Optimal Response

Additionally, each IDIP node sends a copy of the attack report (along with the local responses) to the Discovery Coordinator (Figure 2-7). The Discovery Coordinator can then correlate reports to gain a better overall picture of the situation, and also issue response directives back to individual nodes (Figure 2-8) to either remove an unnecessary response (e.g., firewall filtering rule), or add a response (e.g., firewall filtering rule along an alternate attack path). The Discovery Coordinator is expected to be co-located with the domain's network management facilities, providing the Discovery Coordinator with the network global topology, enabling the selection of the optimal points in the network to block harmful connections.

Figure 2-1 not only shows intrusion detection systems and boundary controllers as IDIP nodes, but also shows that hosts may participate in an IDIP system. Hosts can provide more fine-grained responses as they can trace the intrusion back to the process and user initiating the intrusion from the local host. When an intruder is performing an attack after hopping through multiple hosts, IDIP-enabled hosts allow the intrusion to be traced back through these hosts, which is not possible if only boundary controllers participate in the IDIP system.

Note that allowing hosts to participate in IDIP raises two significant issues for the underlying protocol mechanisms: (1) IDIP neighborhoods may grow to be very large, and (2) some IDIP nodes may be significantly less "trustworthy" than others because they may have a number of vulnerabilities available for an attacker to use. Because neighborhoods may grow very large, IDIP is designed for multicast operation. At the application level, all neighborhood communication is multicast. This second factor implies that some IDIP nodes may be compromised and potentially used against the system. For this reason, IDIP has features that enable it to distinguish less trustworthy components from more trustworthy components.

3. PROJECT ACCOMPLISHMENTS

The following sections detail the project accomplishments, including capabilities developed, IDIP rearchitecting, contributions to CIDF, component integration, demonstrations and experiments, and documentation, as well as a brief description of related work.

3.1 Overall Accomplishments

The major contribution of this project to the Information Assurance program was to provide the primary integrating mechanism for many of the Information Assurance demonstrations and experiments. Through the course of the program, IDIP became the means to easily add new functional capabilities (e.g., INFOCON distribution) that required inter-device cooperation. This was primarily because IDIP was integrated with most security technologies developed or used in the Information Assurance program.

Beyond providing the glue for security component integration, IDIP provided functionality used in multiple experiments and demonstrations. IDIP was central to the first few integrated feasibility demonstrations. The first major program integrated feasibility experiment (IFE) focused on intrusion response. Subsequent experiments that focused on intrusion response using

IDIP included the distributed denial of service (DDoS) and vulnerability assessment experiments. These are described in detail in subsequent sections.

In providing the integration medium for several demonstrations and experiments, several extensions to the core IDIP functionality were developed to improve functionality and robustness of the intrusion detection and response infrastructure. These are described in Section 0.

3.2 Accomplishment Details

This project focused on several related areas of intrusion response:

1. Improving the core IDIP functionality to support demonstrations and experiments planned by the Information Assurance system integrator.
2. Restructuring the core IDIP functionality to enable low-cost integration of varied security technologies.
3. Supporting the Common Intrusion Detection Framework (CIDF) working group in developing a standard format for exchange of intrusion detection and response data.
4. Developing the discovery coordinator concept and features to enable monitoring and control of the intrusion detection and response infrastructure.
5. Refining the IDIP cryptographic mechanisms to enable self-protection of the intrusion detection and response infrastructure.
6. Integration of selected security technologies into the intrusion detection and response infrastructure.
7. Support for Information Assurance program demonstrations and experiments.

The following sections provide details of the work accomplished in each of these areas.

3.2.1 Capabilities Developed

The Adaptive System Security Policies contract provided a number of mechanisms to control detector and response functions of the IDIP agents. The Automatic Response to Intrusion project integrated these mechanisms into the IDIP code base and extended the policy mechanisms to support the needs of various demonstrations and experiments. Among the extensions were -

- Control of vulnerability scanning within an IDIP community. This feature enables the Discovery Coordinator to schedule vulnerability scans across an administrative domain and coordinate this scanning with the detection and response infrastructure to prevent reporting and response to friendly scans.
- Management of firewall and filtering router rule sets.
- Extensions to IDIP detector agent software to enable transport of component-specific information in attack reports. This enables operators to view the additional information available from some detectors that is not otherwise required by the IDIP infrastructure.
- Integration and extension of intrusion detection and response policy mechanisms.

- Development of core Discovery Coordinator components to support system-wide situation assessment, generation of global responses, and management of intrusion detection and response capabilities.
- Improvements to IDIP cryptographic mechanisms to improve system robustness, including use of (1) higher performing (and more readily accessible) software cryptographic algorithms, (2) more robust key distribution algorithms, and (3) X.509 certificates as IDIP credentials.

3.2.1.1 IDIP Vulnerability Assessment Mechanisms

Several IDIP mechanisms both within the IDIP agents and the Discovery Coordinator were developed to support the use of scanners within an IDIP community. These include-

- Discovery Coordinator controlled scheduling of scans.
- Coordination of scans with IDIP agents to avoid reporting friendly scans as attacks.
- Discovery Coordinator processing of vulnerability reports and generation/distribution of new agent policy files.
- Agent policy mechanisms to control IDIP responses for exploits for which the system is not vulnerable.
- Agents can be configured to report, trace, audit, or block on exploits for which the system is not vulnerable.

These new mechanisms allow the system administrator to specify how to respond to friendly scans and how to respond to exploits to which the system is not vulnerable. For example, if the system is not vulnerable to a specific exploit, the policy can be configured to report and trace the attack, but not perform and blocking. [4] provides more detail on this feature.

3.2.1.2 Packet Filtering Rule Management

IDIP policy mechanisms were extended to support generic specification of packet filtering rules for IDIP-enabled response components. These rules are all parameterized by INFOCON and defensive posture to enable automated change of firewall and filtering router rules when either INFOCON or defensive posture changes. These mechanisms provide central management of firewalls and filtering router access control features within an administrative domain. These same mechanisms could also be used to configure filtering rules on end systems through configuring personal firewalls or mechanisms such as Transmission Control Protocol (TCP) wrappers.

One aspect of this development worth mentioning is that with the use of existing IDIP mechanisms, the new capabilities only required 1 day to develop. The existing policy framework and policy distribution mechanisms enabled these changes with only a small amount of additional software.

3.2.1.3 Transport of Component-Specific Information

One of the problems with using a canonical list of attack names (such as those used in CIDF [5]) is that when new attacks are discovered, if the canonical list has not kept pace with the new attacks, there is no way to represent the new attacks. IDIP overcomes this drawback by carrying component-specific attack names and other component-specific fields within the CIDF payload. This component attack name can be used when there is no mapping from the component-specified attack to the CIDF attack codes. Response devices that do not understand this attack code will still be able to respond if the attack class is specified. If the attack class is unspecified, then response devices will trace and report, but not take other actions. Administrators at the Discovery Coordinator can still generate a response based on human understanding of these new attacks.

3.2.1.4 Integrating the Intrusion Detection and Response Policy Mechanisms

The Adaptive System Security Policies project developed a "cost model" approach to determining appropriate responses to cyber attacks. This cost model determines the cost (in terms of system degradation) of the attack and attempts to find a response that costs less than the attack. Cost models were developed for use by the detection, response, and Discovery Coordinator components. These cost models use policies to determine how the intrusion response system should react to various situations. All of these policies are parameterized by INFOCON and defensive posture to enable a different policy set to be used for each INFOCON and defensive posture. The Automatic Response to Intrusion project integrated these mechanisms by enabling centralized management of these cost models at the Discovery Coordinator.

The initial policy components were available for IFD 1.2. These were modified and extended as new requirements for controlling the intrusion detection and response system emerged from requirements for each demonstration and experiment. The core policy framework enabled most of the policy processing and distribution functions to be shared by the various policy components. This strategy enabled easy addition of new policy elements.

The policy mechanisms enable an administrator to specify unique policies for each interface of each IDIP-enabled component or to use a generic policy applicable across the entire administrative domain. The administrator may also specify that some components use the generic policy, while others have a unique policy. Policy mechanisms for IDIP agents include rules for event aggregation, false positive suppression, repeated alert suppression, host-based response (e.g., kill process or disable user account) control, network response (e.g., block web access) control, and vulnerability assessment scanning. Discovery Coordinator policies control the generation of system-wide responses.

The policy mechanisms are described in detail in [4].

3.2.1.5 Discovery Coordinator Development

At the beginning of this project, the Discovery Coordinator was a location to receive alerts with a primitive command line display. Making the IDIP infrastructure usable for the varied demonstrations and experiments in the Information Assurance program required that several features be added to the initial Discovery Coordinator.

Our strategy was to produce relatively lightweight independent processes, where each process provides a specific Discovery Coordinator feature. This strategy enabled easy modification of Discovery Coordinator features and addition of new features because these independent processes have very few dependencies. Interactions between Discovery Coordinator processes generally occur over the standard IDIP message service, which allows process to subscribe to any of the message types desired. This publish-subscribe approach helps maintain low coupling between processes.

The Discovery Coordinator receives reports from each detection and response component involved in reacting to intrusion events. Because the Discovery Coordinator is expected to be co-located with network management mechanisms, the Discovery Coordinator can use the reports (which describe the attack, provide the attack path, and detail the response taken along the attack path) to develop an optimal response for the administrative domain. The component that determines the optimal response uses the Discovery Coordinator policy mechanisms to provide user control over these global responses.

Beyond generating global responses, Discovery Coordinator processes were developed to (1) combine related reports to avoid taking multiple responses to multiple reports of the same event, (2) forward policy files to IDIP agents on either initialization or when policy changes are made, (3) distribute changes to INFOCON and defensive posture, (4) interface to the public domain Scotty network management system, (5) provide a graphical user interface (GUI) for editing IDIP policy, (6) interface to the Information Assurance program's Cyber Command System, (7) modify response policy based on results of vulnerability scans, (8) schedule vulnerability scans, and (9) transform CIDF-formatted IDIP reports into other formats for consumption by other tools.

Several interfaces were developed to support integration with other components that attempt to provide a global view of network attacks. Interfaces were developed to support integration with U.C. Davis's Graphical Intrusion Detection System (GrIDS), Intrusion Detection Message Exchange Format (IDMEF) defined by Internet Engineering Task Force (IETF) [6], MountainWave's tools, and Rome Lab's interface to Oracle used by their Automated Intrusion Detection Environment (AIDE) [7] component.

The Scotty network management system provides the primary user interface. IDIP extension to the Scotty's user interface provides notification of attacks by coloring network components on Scotty's network map as the Discovery Coordinator receives attack reports. Attack detectors, victim, source, and responders are each colored differently to help the administrator understand how the intrusion detection and response system reacted to the attack. Other extensions to the GUI allow the administrator to "undo" the automated responses taken by IDIP-enabled components, display the network map for older attacks, and view the IDIP report details. The IDIP extensions to Scotty also allow other Discovery Coordinator processes to issue Simple

Network Management Protocol (SNMP) messages as a response mechanism. For example, this feature enables the Discovery Coordinator to issue interface shutdown commands to SNMP-controlled devices. This extends the reach of the intrusion response system beyond IDIP-enabled devices to those devices that response to SNMP messages.

The interface developed for the Cyber Command System enables that component to perform the same features available to other Discovery Coordinator processes: (1) issue a response directive, (2) undo a response, (3) receive IDIP reports, (4) change INFOCON and defensive posture, and (5) modify IDIP policies.

On additional feature IDIP supports is distribution of Discovery Coordinator mechanisms. The Discovery Coordinator is provided a multicast address, and the IDIP configuration files specify the set of IP addresses that are mapped to that multicast address. IDIP agent and Discovery Coordinator applications use the Discovery Coordinator multicast address to communicate with Discovery Coordinator processes, and the IDIP multicast mechanisms provided reliable, secure message delivery to those Discovery Coordinator processes that have subscribed for those messages. This enables redundancy to be used to increase Discovery Coordinator survivability and spreading the Discovery Coordinator load across multiple machines to reduce processing load.

3.2.1.6 IDIP Cryptographic Protection

To protect IDIP messages from adversary exploitation, IDIP cryptographic mechanisms were developed in the initial Dynamic Cooperating Boundary Controllers effort. Those initial mechanisms were based on the Fortezza cryptographic card and provided basic privacy, authentication, integrity, and key distribution services for IDIP messages. However, the Fortezza implementation was sufficiently slow to adversely impact IDIP's ability to react to attacks that generated a large number of reports or attacks through an already busy boundary controller. Another drawback of the initial implementation was the use of a custom certificate format. [8]-[12] provide the details of the IDIP cryptographic protocols.

To overcome these performance and interoperability issues, this project made two major improvements to the IDIP cryptographic mechanisms: (1) implementation of cryptographic services using a public domain cryptographic library [13] and (2) use of the standard X509v3 certificate format for IDIP credentials. A third feature developed was a mechanism to support administrative domains where some components are less trustworthy than other components.

Table 2-1 shows the cryptographic algorithms supported by the Fortezza and software cryptographic mechanisms.

Cryptographic Feature	Initial Fortezza Implementation	Current Software Cryptographic Implementation
Data Privacy	Skipjack	DES
Data Integrity	SHA-1	SHA-1
Digital Signature	DSA	DSA
Key Exchange	KEA	El Gamal (Modified Eric Young's Diffie-Hellman)

Table 2-1. IDIP Supported Algorithms

The software cryptographic functions provided dramatic performance improvements over the Fortezza algorithms as shown in Table 2-2. The poor performance of the Fortezza-based implementation caused some instability in IDIP operation in IFD 1.1. The software cryptographic mechanisms improved performance sufficiently that the stability problems did not reappear in later demonstrations and experiments.

System	Operation	Fortezza		Software Crypto	
		Message Size		Message Size	
		1K	.25K	1K	.25K
Solaris	HMAC-SHA	168 ms	158 ms	2 ms	2 ms
	Encrypt	122 ms	119 ms	2 ms	1 ms
	Decrypt	120 ms	116 ms	2 ms	1 ms
	Sign	191 ms	177 ms	152 ms	152 ms
	Verify	255 ms	239 ms	299 ms	301 ms
BSDI	HMAC-SHA	159 ms	152 ms	2 ms	2 ms
	Encrypt	121 ms	119 ms	1 ms	1 ms
	Decrypt	114 ms	112 ms	1 ms	1 ms
	Sign	178 ms	170 ms	46 ms	47 ms
	Verify	250 ms	235 ms	92 ms	93 ms

Table 2-2. Cryptographic Mechanism Performance Comparison

The major changes required to support X509 certificates were in the software implementing the Neighborhood Key Information Distribution (NKID). There were a small number of extensions required to NKID to enable support for both the IDIP and X509 credential formats.

To support less trustworthy IDIP nodes (e.g., client workstations), the notion of a subneighborhood was added to the IDIP messaging services and the cryptographic mechanisms. A subneighborhood can be established within an IDIP neighborhood to separate messaging required by the more trusted components from communication with all neighbors. Each neighborhood can support multiple multicast groups, each representing a collection of components that are equally trustworthy.

A final issue with the initial cryptographic services addressed by this effort was the key distribution protocol (NKID). The NKID initial version relied on a neighborhood captain to distribute keys for the neighborhood. This approach proved unreliable in certain attack scenarios.

A better strategy was developed where each node in a multicast group generates its own transmit key. This removes a single point of failure and a source of software complexity from our initial implementation.

3.2.2 IDIP Rearchitecting

The initial IDIP code base, developed for the Dynamic Cooperating Boundary Controllers project, was used to demonstrate the feasibility of automated response. The code was used to integrate a small number of components: a custom router, firewall, and detector. Supporting the larger set of components required by the Information Assurance program, coupled with the continued uncertainty of which components would be required next, led to the conclusion that the initial IDIP software architecture would be too costly to maintain. A new software architecture for our IDIP code was developed that reduced the amount of component-specific software. This architecture is depicted below in Figure 3-1 for the generic response component. The architecture for detection components is similar.

This new architecture isolates most of the IDIP functionality from component-specific features enabling adding new features at low cost and enabling changes to the on-the-wire encoding without modification of the core software functionality. The new architecture also reduced the amount of software needed to integrate new components into the IDIP infrastructure. This greatly reduced the cost and risk of integrating with new components such as NetRadar and Sidewinder. This architecture also isolated operating system specific functions into a separate library. Porting to a new operating system requires modification to this one file to provide the operating system features required for the core IDIP functionality.

The following sections describe the architecture for IDIP response, detection, and Discovery Coordinator applications. [14] has more detail on the architecture and specific features of these three types of applications.

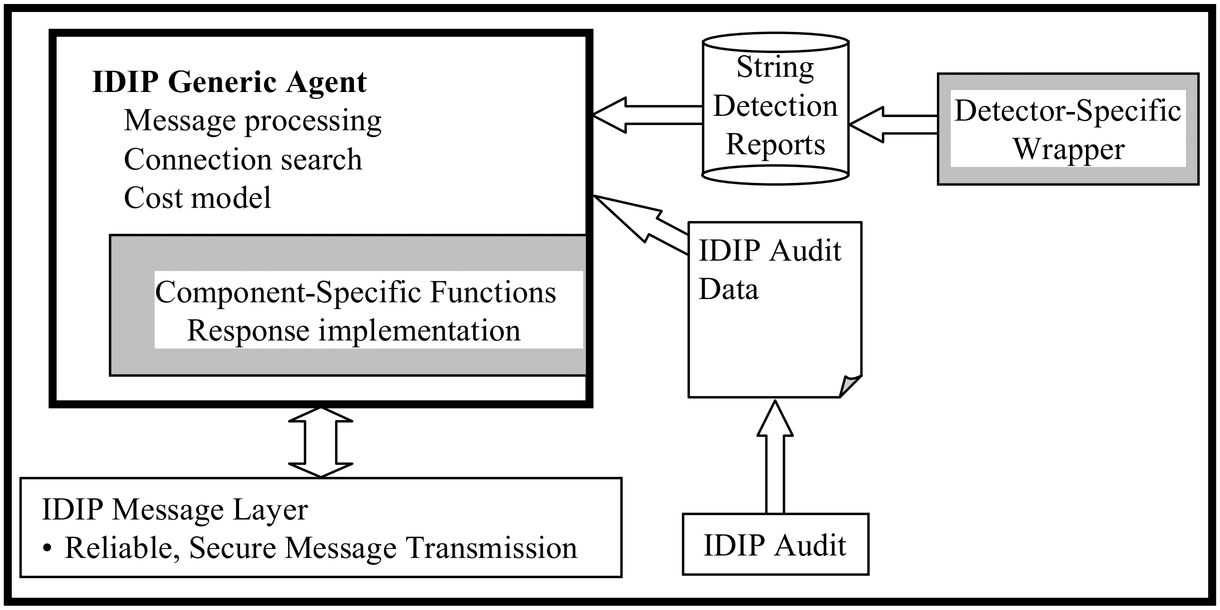

Figure 3-1. IDIP Generic response agent architecture

3.2.2.1 IDIP Response Applications

Figure 3-1 shows both detection and response functions. IDIP response components can also be used as detectors. Most IDIP responders are access control components. Attempts to gain unauthorized access are an indicator of attempted attacks. A component-specific detector wrapper process writes intrusion inputs to an event report file in the IDIP string format. Generic functions in the IDIP agent read event reports from the file and apply the intrusion detection policy mechanisms to determine if the event (or accumulation of events) warrants generation of an attack report. Attack reports initiate the response engine's actions. The response engine uses the response policy to determine (1) what local responses (e.g., kill a local process or install a packet filtering rule) are required, (2) whether to trace the attack, and (3) whether a report should be sent to the Discovery Coordinator. Only the mechanisms for handling the local response actions are component-specific. The remainder of the software is common between response components. When messages are sent, the IDIP message layer provides reliable secure message delivery.

Response agents also respond to trace messages from other IDIP components. On receipt of a trace message the IDIP agent searches the IDIP audit trail of recent connections for matches with the packets described by the IDIP trace message. The IDIP audit mechanism provides generic functions based on a public domain monitoring library [15]. These mechanisms monitor network traffic through the device and record in a network audit trail the address and port information for the connections and datagrams that pass through the device. If it is determined that the attack may have come through the device, the agent uses the policy mechanisms to determine the local response to take, and then forwards the trace message to its neighbors along the attack path and sends a report to the Discovery Coordinator.

3.2.2.2 IDIP Detector Applications

The generic IDIP detector uses the same detection interface as the response components. The major difference is that detectors do not respond to trace messages. Because of this, detector agents do not need the IDIP audit functions or the component-specific response functions.

The string API used by both detection and response components was originally developed to enable easy integration of detectors. This API enabled independent development of component-specific detector wrappers. Developers of these wrappers do not require the full IDIP software base, just a simple string validation process that was developed to determine whether the IDIP software would be able to parse the strings provided by the wrapper. This API was first used to integrate RealSecure, which took only 10 hours demonstrating the utility of this approach.

3.2.2.3 Discovery Coordinator Applications

Because each Discovery Coordinator application has a unique purpose, they use less shared functionality than the generic agents do. Discovery Coordinator applications use the same interface to the message layer as the generic agent application.

Because there was little definition of what should exist at the Discovery Coordinator, the objective was to provide a flexible environment. This led to the following features: (1) support for a publish-subscribe interface that enabled loosely coupled message sharing, (2) use of multicast addressing for the Discovery Coordinator enabling the Discovery Coordinator functions to be split among multiple hosts with all applications having all visibility of all messages, (3) support for multiple correlation components enabling components with different strengths to be used concurrently, and (4) minimal set of required processes. The only required process at the Discovery Coordinator is the policy distribution process for distributing policy and configuration data to IDIP agents as they initialize. Running the local GUI or the interface to the Cyber Command System also requires the use of the local attack aggregation process. Other processes (e.g., global response generation) are only used if needed by the current demonstration or experiment.

This strategy of minimizing the coupling between functions at the Discovery Coordinator enabled easy integration of new features and modification of existing capabilities. Integrating with other tools was accomplished by adding a simple adaptor process to convert the incoming alerts or response messages to the format required by the tools. For example, integrating with AIDE involved converting the CIDF messages to a string format used by Rome Lab's interface software for insertion into an Oracle database. This integration only required a couple of days effort, most of which was spent determining the semantics of the database fields and the mapping to CIDF constructs. Although this interface was never used, the development effort illustrates how the Discovery Coordinator architecture supports ease of integration.

3.2.3 Common Intrusion Detection Framework (CIDF)

Although not part of the original project plan, this project contributed significantly to the CIDF effort, including (1) developing approaches and Common Intrusion Specification Language

(CISL) [5] constructs for communicating response requests and actions; (2) developing an approach to modifying CISL messages without losing the original data or signature (e.g., because of translation required from connections that pass through firewall proxies and network address translation devices); (3) developing the CIDF message layer [16]; (4) contributing to CISL binary encoding rules; (5) contributing to removal of CISL specification ambiguities; (6) developing a CIDF implementation of IDIP for use in integrating components for the CIDF demonstration; and (7) serving as integrator for the CIDF demonstration.

The CIDF working group held two separate interoperability tests that this project supported. Our initial CIDF application layer implementation was used for CIDF interoperability testing at U.C. Davis June 15-17 1998, providing one of the three implementations of the CIDF encoding. These were each determined to be interoperable after some minor adjustments during the week. The second major test also served as IFE 2.2 in June 1999, and is described in more detail in Section 0.

3.2.4 Component Integration

In support of the Information Assurance program demonstrations and experiments, IDIP was integrated with selected COTS products and research prototypes. Table 3-1 shows the set of detection and response components integrated through the life of the project.

Beyond this set of components, CyberCop Scanner [35] was integrated to help assess the issues involved in integrating vulnerability scanners. [4] details how the integrated system operates. The scanner integration required some changes to the IDIP core software, including extensions to the policy mechanisms to support control of scanning and control of responses to friendly scans. Detection and response agents also require knowledge of when scans are occurring so that they can determine whether a scan is friendly or not. Most of the new software was developed to configure the scanner and process the results. The two primary uses for scan data are (1) to detect unexpected changes in vulnerability results, which may indicate that the scanned component has been compromised and (2) to modify IDIP response policies so that the responses are less harsh for those attacks for which the target component is not vulnerable.

Response Components	Intrusion Detection Systems	Correlation Components
NAI Gauntlet Internet Firewall™ (multiple versions) [17] (product transitioned from NAI to Secure Computing)	Net Squared Network Radar [26]	U.C. Davis GrIDS Prototype [32]
Secure Computing Corporation Sidewinder™ Firewall [18]	SRI EMERALD Basic Security Module (BSM) and EMERALD File Transfer Protocol (FTP) Monitors Prototypes [27]	Stanford Complex Event Processing (CEP) Correlator [33]
Linux Router NetFilter and IP Filter [19] [20]	Oregon Graduate Institute StackGuard [28]	Silicon Defense Corroborator
NAI Labs ARGuE Prototype [21]	ORA Common Object Request Broker Architecture (CORBA) Immune System Prototype [29]	Rome Labs AIDE system [7]
NAI Labs Multi-Protocol Object Gateway Prototype [22]	NAI CyberCop™ Server and CyberCop Monitor (no longer supported by NAI)	Mountain Wave Recon system [34]
Firewall Toolkit [23]	NAI VirusScan [30]	IDIP's mechanism for merging multiple reports of the same event
TCP Wrappers [24]	Internet Security Systems RealSecure™ [31]	SRI EMERALD IDIP monitor
NAI Labs Generic Software Wrappers Prototype [25]	Custom MD5-based file modification detector	
Solaris process and user account management mechanisms	Custom Oracle log based detector	
	Custom promiscuous mode detector	
	Custom BSM log-based detector	

Table 3-1. Components Integrated with IDIP

™ Gauntlet is a registered trademark of NAI. Sidewinder is a registered trademark of Secure Computing Corporation. CyberCop is a registered trademark of NAI. RealSecure is a registered trademark of ISS.

Most of these components required very little, if any, modification of core IDIP functionality to achieve full functionality. However, for Multi-Protocol Object Gateway, several new responses were desired to support the object-based access control mechanisms of the component. These new responses were integrated with the policy mechanisms to enable control of these responses from the Discovery Controller and with CISL to enable description of the responses in reports to the Discovery Coordinator. The following lists the Multi-Protocol Object Gateway (MPOG) responses.

- Deny Role
- Deny Principal
- Deny Principal in Role
- Deny Client
- Deny Principal from Client
- Deny Service
- Deny Service from Client
- Deny Server
- Deny Service on Server
- Require Use of Security Protocol

3.2.5 Information Assurance Demonstrations and Experiments

Initially, the Information Assurance program hosted periodic Integrated Feasibility Demonstrations (IFD) to demonstrate the extent of integration of technologies under development by the Information Assurance program. After the initial IFDs, the program began hosting periodic major experiments that involved many of the technologies under development. The following sections describe this project's participation in these events.

3.2.5.1 IFD 1.1

IFD 1.1 was a demonstration of the capabilities of Information Assurance program technologies very early in the program before inter-project integration could occur. This project demonstrated the integration of components that were developed under the Dynamic Cooperating Boundary Controllers contract, plus Gauntlet, CyberCop Server, and GrIDS. The primary result of the effort involved in this integration was the beginning of the IDIP software restructuring to enable lower cost component integration.

3.2.5.2 IFD 1.2

IFD 1.2 was the initial effort at integrating other Information Assurance program components. Beyond the integrated components from IFD 1.1, NetRadar and Sidewinder were integrated. This was the first integration of the Discovery Coordinator system-level response mechanisms. A SNMP trap mechanism was developed to enable the Discovery Coordinator to send the initial version of the Cyber Command System these system-level responses. This IFD also used a red team from GTE to begin testing the ability of the system to survive attack. The IDIP infrastructure was able to track the GTE red team progress and display these alerts at the Discovery Coordinator. A lesson learned from the red team exercise in this IFD was that a mechanism was needed to throttle detection reports, particularly repeated reports of the same events. By scanning an entire network, the red team was able to clog the IDIP infrastructure with intrusion reports of the scan. These thousands of reports all represented one event, but were treated as separate events by the detectors and hence by the IDIP infrastructure.

3.2.5.3 IFE 2.1

Beginning in the 1999, the Information Assurance IFDs were transformed into Integrated Feasibility Experiments (IFE). These IFEs were intended to explore problems in information assurance. However, IFE 2.1 occurred before the experimentation approach was developed and was really just another IFD.

For this IFE, a number of components were integrated, along with some improvements made to the core IDIP software. IDIP software improvements integrated included the new message layer software, key distribution software, detection and response policy mechanisms, generic detection and response processes, string API, and new Discovery Coordinator components. Integrated components included EMERALD, NetRadar, NAI Labs UNIX wrappers, ARGuE, MPOG, NetRadar, CyberCop Server, Gauntlet, RealSecure, and the Cyber Command System. During the IFE, an experiment in Cyber Command System flexibility was accomplished by using the flexibility of the IDIP infrastructure to develop some additional IDIP capabilities to cordon off a subnet. This development became part of the core demonstration performed for high-profile visitors. The IDIP infrastructure provided the detection and response mechanisms necessary for support. One feature that was available in IDIP only during IFE 2.1 was the capability to attack the attacker's machines. The mechanism was used as part of the red team exercise during IFE 2.1, again using the GTE red team. Once the team established a beach head on one of the systems outside the primary firewall, the IDIP software launched a counter-attack against the compromised machine to disable it from further communication. This frustrated red team efforts because they did not detect the counter attack, and were unable to continue their attacks.

3.2.5.4 CIDF Demonstration (or IFE 2.2)

The June 1999 CIDF demonstration was used as a test of CISL maturity. Specific objectives were to-

- Test ability to create interoperable CIDF components.
- Validate utility of CISL as a language for enabling correlation
- Demonstrate some simple correlation that shows value-added created by sharing analysis results. The focus was on reducing the number of false positives and raising the number of detections through analysis of detection results across the system.

Figure 3-2 shows the configuration for the CIDF demonstration. Each detection component in the demonstration was wrapped with an IDIP wrapper and integrated via IDIP into the system. Correlation components were integrated at the Discovery Coordinator where all attack reports were collected and distributed to these correlators. The following components were integrated with IDIP to support the demonstration.

- Detectors
  - SRI EMERALD BSM Analyzer
  - ORA CORBA Anomaly Detector
  - Oregon Graduate Institute StackGuard
  - Net^2 NetRadar
  - NAI CyberCop Server
  - Internet Security Systems (ISS) RealSecure
- Correlators
  - Silicon Defense Corroborator
  - Stanford Complex Event Processing (CEP) Correlator

The demonstration successfully accomplished the objectives through successful system integration and development of algorithms by Stanford and Silicon Defense for using corroboration to reduce false positives. Using heterogeneous detectors and the IDIP infrastructure that collects attack reports at the Discovery Coordinator supported improved detection rates.

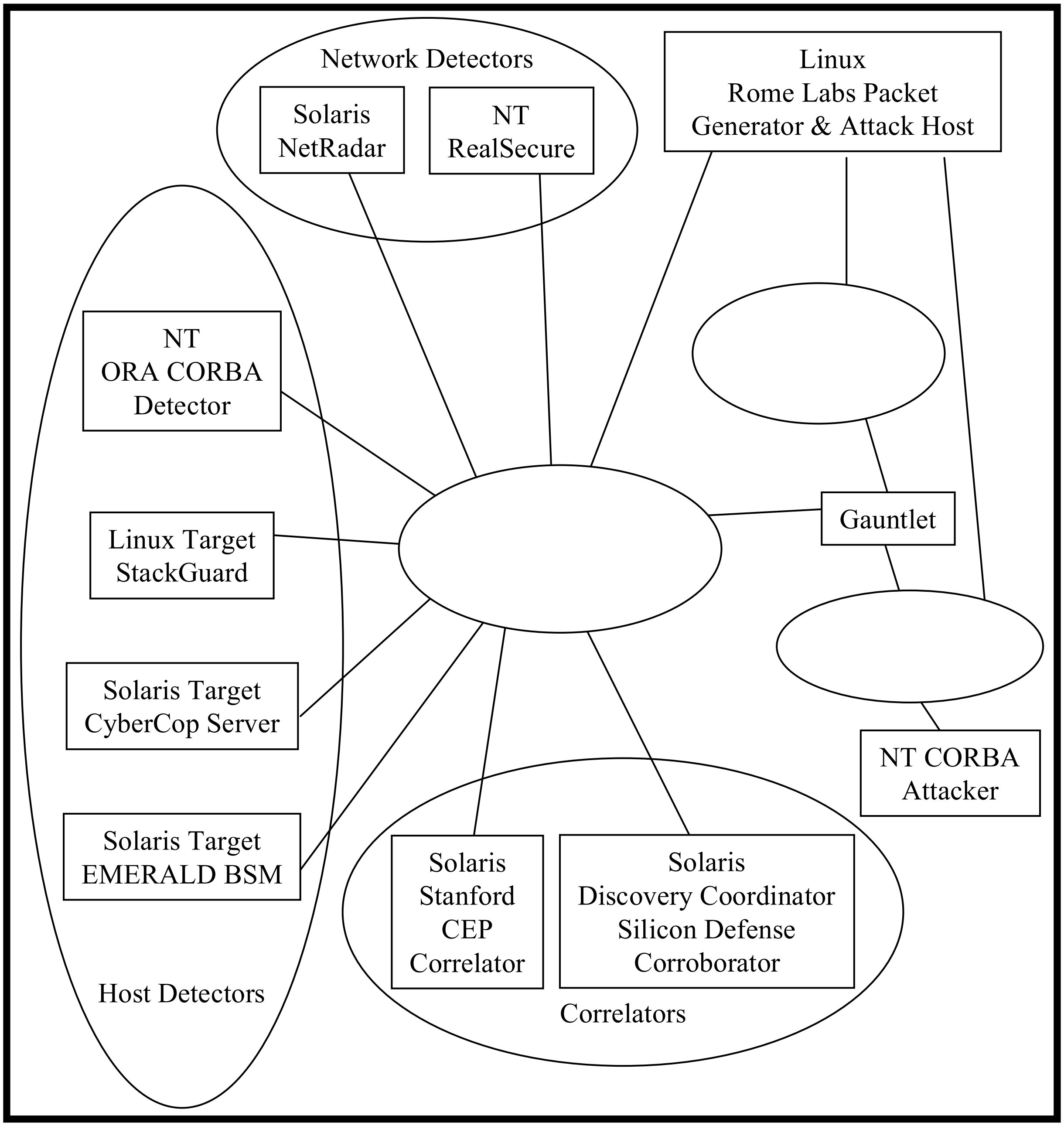

Figure 3-2. CIDF Demonstration Configuration

3.2.5.5 IFE 2.3

IFE 2.3 was the first major Information Assurance program-level experiment. The objective of the experiment was to assess the viability of automated response as embodied in the IDIP implementation. Unfortunately, flaws in the application used in the demonstration enabled the red team from Sandia to sniff the user's login and database passwords, and using those behave as a normal user achieving their objectives undetected. A follow-on run of this experiment did not fix the application, but rather tried to improve the detection capabilities to be able to see the red team activities in spite of the red team being able to gain access to user login and database passwords. The new detection capabilities were able to detect the red team activities; however, the response policy used during experiment execution was not appropriate to the job of stopping the red team. This policy mis-configuration allowed the red team to proceed, just at a slower pace. [36] includes details of the IFE and IFE rerun. Unfortunately, the only conclusions one could derive from IFE were that (1) a sufficiently poorly conceived security design results in flaws that can be easily exploited without detection and (2) poor configuration of the response mechanisms cannot be expected to stop the adversary.

IDIP-enabled components involved in the IFE included CyberCop Server, EMERALD BSM Monitor, EMERALD FTP Monitor, NetRadar, RealSecure, CyberCop Monitor, Gauntlet 5.0, Sidewinder, MPOG, IP Filter, NAI Generic Software Wrappers, Discovery Coordinator, System Management and Administration for Remote Trusted System (SMARTS) (integrated through the Cyber Command System), and the Cyber Command Center. New IDIP features for this IFE included some refinements to the IDIP software architecture, SNMP-based responses that disable Ethernet ports on either hosts or switches, updated cryptographic mechanisms, new startup software, new IDIP audit mechanisms, and CIDF implementation of IDIP.

For the IFE 2.3 rerun, some new detection and response capabilities were developed and integrated. New detection capabilities included-

- ifstatus, which detects connection of a sniffer to the local host
- Oracle log processing
- NetRadar strings, which searched for disallowed commands
- NetRadar duplicate FTP connection detection
- EMERALD BSM disallowed commands
- MD5-hash, which detects modified files
- NAI Generic Software Wrappers disallowed command detection

A BSM reader was also developed that can detect disallowed command usage. This was developed as a backup to EMERALD and Wrappers integration, but was not needed for the IFE.

The new response features integrated were host-based responses that enable an IDIP-enabled host to kill processes and user sessions, and disable user accounts. Unfortunately, killing the process instead of the user login session was used. This just kills the offending shell command, which has generally completed its malicious task by the time detection occurs. Killing the user session and disabling the compromised user account would have been a more appropriate response and was recommended for experiment execution, however, it was not used. Had this policy been used, it is unlikely that the red team would have been able to accomplish their objectives so easily.

3.2.5.6 IFE 3.1 Integration

For IFE 3.1, major changes to the Discovery Coordinator were made to enable Cyber Command System control of IDIP policy mechanisms. In support of this integration, additional GUI components were also developed to enable users at the Discovery Coordinator to modify IDIP policy features. The GUI components provided both editing capabilities for detailed policy modifications, plus some slide bar features to allow global changes of policy values and detector throttling. Other controls provide access to INFOCON and Defensive Posture, a software-controlled policy switching mechanism. A final GUI feature developed was an "attackometer" that shows how much attack activity is currently being experienced by the administrative domain.

3.2.5.7 Distributed Denial of Service Experiment

The Information Assurance program distributed denial of service experiment was designed to determine the ability of the IDIP infrastructure to respond to distributed denial of service attacks. In an attempt to validate the results, BBN repeated the experiment in the DARPA Technology Integration Center. This project supported that effort by providing trouble-shooting of problems they experienced in their lab setup.

The experiment results indicate that (1) the IDIP message layer does in fact survive the flood attacks even when all other communication fails; and (2) the automated response can effectively stop the attack allowing the application to proceed. Details on the experiment configuration, execution, and detailed results are in [37].

3.2.6 Documentation

This project contributed to a number of technical reports detailing the IDIP architecture (ref. [1]), application protocol (ref. [2]), message layer protocol (ref. [3]), key distribution protocol (ref. [8], [9], [10], [11], [12]), and cryptographic privacy and integrity protocols. The IDIP architecture, application protocol, and message layer protocol documents were started in this contract, but completed in two other DARPA contracts: Active Networks Intrusion Detection and Response and DDoS Tolerant Networks. The key distribution and cryptographic protocols were originally developed under the Dynamic Cooperating Boundary Controllers contract, but were updated in this effort. In addition, this project contributed to three papers: (ref. [4], [14], and [37]).

The other major documentation effort performed by this contract was the development of HTML-based user documentation. This documentation was developed to describe how to compile, configure, and run the core IDIP applications, including descriptions of how the policy mechanisms can be used to control the intrusion detection and response infrastructure. This HTML-based documentation was subsequently updated under the Active Networks Intrusion Detection and Response and DDoS Tolerant Networks, and is distributed with the IDIP software. This documentation was tested by having a summer intern compile, configure, and run a set of IDIP applications having no training on IDIP. This successful test indicates that the documentation is suitable for use by knowledgeable developers.

3.3 Related Work

This project was able to develop a large number of capabilities through synergy with two other concurrent efforts: Adaptive System Security Policies and Information Assurance Integration. All three projects developed components integrated into the shared IDIP code base. The demonstrations and experiments in the Information Assurance program provided feedback that was used to make the mechanisms developed in each of these three projects more robust.

The Adaptive System Security Policies project focused on implementation of the policy mechanisms used to ensure that an IDIP-enabled system does not cause self-inflicted damage worse than effects of the attack being blocked. The Automatic Response to Intrusion project developed the management mechanisms to make better use of these policy mechanisms and executed the demonstrations and experiments used to test these mechanisms.

The Information Assurance Integration project focused on how the intrusion detection and response infrastructure interfaced to the Cyber Command System. The Automatic Response to Intrusion project used the results of this integration as input to refinements to the intrusion response infrastructure development.

This project also developed documentation of the IDIP application and message protocols, which was subsequently completed in the Active Networks Intrusion Detection and Response and DDoS Tolerant Networks projects.

The results of this contract also provided a starting point for the Multi-Community Cyber Defense project, which expanded on the issues identified in this project related to intrusion response across multiple administrative domains.

4. CONCLUSION

The focus of this project was to develop an intrusion detection and response infrastructure in support of Information Assurance demonstrations and experiments. Using this technology in environments subjected to red team attack provided significant insight into how to strengthen the infrastructure to better withstand attack and support better responses.

Lessons-learned. The integration and experimentation process led to several lessons learned.

a. **Experiment design.** The initial IFE 2.3 failed to produce useful results for a number of reasons. The introduction of an easily penetrable application in a highly secure environment yielded a highly vulnerable system. The security measures could not overcome what were obvious application deficiencies. Part of this was driven from a desire by some developers to not make the red team’s job too difficult. This viewpoint was also apparent in the IFE 2.3 rerun where the response policies used were less than optimal to avoid making the red team’s job too difficult. A second problem in the experiment implementation was that the implementers made assumptions about the capabilities of selected components that were not based on fact, but rather on conjecture. The result was that the experiment did not test what it was intended to test. The lesson here is that the experiment detailed design must involve researchers and developers sufficiently knowledgeable of the technologies involved who can help assess whether the experiment will yield the desired results.

b. **Intrusion detection and response effectiveness.** In spite of problems with IFE 2.3, the rerun did show that when the attacks are detected, responses can affect red team behavior. Had the correct responses been used, it is likely that the automated intrusion response infrastructure would have significantly increased the red team's cost, however, the experiment implementation failed to test this hypothesis. The DDoS experiment show that automated response can effectively stop the attack before any application degradation is noticed by the user, again providing evidence that automated response is an effective capability when applied correctly.

c. **Low-cost integration strategies.** The effort spent rearchitecting the IDIP software resulted in an infrastructure that enabled low-cost component integration, feature modification, and development of new capabilities. The large amount of software shared between components, plus the simple API for integrating detectors provided most of the cost savings for integration of detection and response components. The initial integration of NetRadar with this API took only 8 hours. The major effort in integrating is determining how the detector's events map into the CISL-defined events. To add a feature for all detection and response components required adding the feature once. The loosely coupled messaging architecture built around a publish-subscribe paradigm enabled easy insertion of new processes to perform new functions.

d. **Simple agent framework.** The fact that almost every component in the Information Assurance demonstrations hosted an IDIP agent enabled easy insertion of new globally available features. The simple publish-subscribe agent framework allowed new features to be added without affecting existing features. Adding the capability to spread INFOCON throughout a system was simply a matter of adding an INFOCON publisher at the Discovery Coordinator and subscribing to the message at each IDIP agent. Adding management features and adding the vulnerability assessment tool management feature also required minimal effort.

IDIP attributes. The mechanisms developed support the original IDIP requirements, as well as providing better control over IDIP response. This project validated that IDIP met the desired attributes set out as system objectives at the start of the project.

a. **Real-time response.** The IDIP response components use simple cost-benefit models to determine an appropriate short-lived response that attempts to eliminate the attack quickly. The algorithm can be made as efficient as standard routing algorithms. The short-term response provides time for more sophisticated algorithms to determine better response.

b. **Multiple administrative domains.** The mechanisms defined allow each administrative domain to operate autonomously with minimal knowledge of remote domains. The issues involved in supporting sanitization of requests and adjusting responses from other domains to allow minimally cooperating domains to respond to intrusions that cross domain boundaries were identified. A subsequent project – Multi-Community Cyber Defense – developed the mechanisms necessary to support this type of operation across administrative domain boundaries.

c. **Minimal system performance impact.** After initialization, IDIP consumes minimal network bandwidth when there is no attack activity. Only infrequent keep-alive messages are used to maintain the neighborhood state. During attacks, use of multicast operation and relatively compact messages minimizes affects on network resources. IDIP-invoked filtering of network traffic, which can cause boundary controller performance degradation, is designed to be relatively short-lived, so that parts of the network far removed from the attack source have only short-term affects once the Discovery Coordinator has implemented the optimal system response. The IDIP mechanisms also only place responses along the attack path (or for Discovery Coordinator-generated responses, along alternate paths available to the attacker).

d. **Operating while the system is under attack.** The use of a lightweight, secure, reliable User Datagram Protocol (UDP) for communication reduces the effects of attacks on IDIP. This was demonstrated during the DDoS experiment.

e. **Autonomous response.** The mechanisms defined allow each IDIP node and each Discovery Coordinator to independently determine their responses based on the IDIP messages and policy parameters.

f. **Scalability.** The use of IDIP "neighborhoods" (see Section 2) limited the knowledge required of each IDIP component enabling easy growth of the IDIP system. IDIP components only have to know about other IDIP components that are nearby, plus their discovery coordinator. This reduces the management required for each component. IDIP has been used with moderate sized neighborhoods such as the one used for IFE 2.3, and the effort to add more nodes into a neighborhood or to add a new neighborhood is roughly linear in terms of configuration effort and performance impact on the network.

g. **Robustness.** IDIP was able to continue intrusion response operation even during attacks that flooded the network or slowed down IDIP components. Additional work, however, could further improve this robustness.

h. **End-user transparency.** No application changes were required to allow IDIP operate within a system. IDIP uses minimal network resources. When the system is under attack, IDIP network utilization increases to support tracing and centralized reporting, but the number and size of messages is still relatively small.

i. **Simplicity.** The IDIP application layer was particularly simple to implement and integrate with the IDIP component prototypes. The use of a simple UDP-based protocol enabled quick development of the IDIP message layer. The primary complexity in IDIP is the key management functions required to provide IDIP self-protection.

j. **Protection against spoofing.** IDIP cryptographic services provide protection against the spoofing of IDIP components through authentication and integrity mechanisms for IDIP messages. A standard keyed hash algorithm is used to support authentication within each IDIP neighborhood. The cryptographic mechanisms also support replay protection.

k. **Compatibility with multiple encryption technologies.** The initial IDIP implementation used Fortezza hardware, however, IDIP's cryptographic mechanisms are algorithm independent enabling integration of additional cryptographic algorithms to support varying user requirements. The current cryptographic are shown in Table 2-.

Summary. This project produced a robust infrastructure for intrusion detection and response and validated the value of automated response in reaction to cyber attack. This infrastructure was integrated with a number of components in support of Information Assurance program demonstrations and experiments. These demonstrations and experiments aided in refining the requirements for the infrastructure and helped test aspects of effective automated intrusion response. In the demonstrations and experiments, the IDIP protocol and this project contributed much of the security technology integration accomplished in the Information Assurance program. Beyond development of this infrastructure, this project also made significant contributions to the CIDF effort, a pre-cursor to the IETF's IDMEF.

The IDIP software demonstrated that automated response can be effective in stopping attacks.

- The IFE 2.3 rerun showed that automated response technology has promise in providing protection against adversary attack provided the combination of prevention and detection mechanisms do not provide the adversary with easy undetectable access to critical system resources. New IDIP-related mechanisms would be useful for handling aggregated attack data and to enable the IDIP agents to be more effective when the Discovery Coordinator is not active.
- The DDoS experiment also showed that automated response is very effective in reacting to DDoS attacks, and can stop the attack before the user notices any application degradation.

5. REFERENCES

[1] D. Schnackenberg, T. Reid, K. Djahandari, B. Wilson, "Cooperative Intrusion Traceback and Response Architecture (CITRA)", NAI Labs Report #02-008, February 2002.

[2] K. Djahandari, B. Wilson, J. Thorpe, D. Schnackenberg, T. Reid, "Intruder Detection and Isolation Protocol (IDIP) Application Layer Protocol Definition", NAI Labs Report #02-006, February 2002.

[3] K. Djahandari, B. Wilson, J. Thorpe, D. Schnackenberg, T. Reid, "Intruder Detection and Isolation Protocol (IDIP) Message Layer Protocol Definition", NAI Labs Report #02-005, February 2002.

[4] D. Schnackenberg, H. Holliday, R. Smith, K. Djahandari, D. Sterne, "Cooperative Intrusion Traceback and Response Architecture (CITRA)", Proceedings of the Second DARPA Information Survivability Conference and Exposition, San Diego, June 2001.

[5] Rich Feiertag, Cliff Kahn, Phil Porras, Dan Schnackenberg, Stuart Staniford-Chen, Brian Tung, "A Common Intrusion Specification Language", June 1999

[6] Intrusion Detection Message Exchange Format (IDMEF) ….

[7] Automated Intrusion Detection Environment (AIDE) ACTD. http://www.if.afrl.af.mil/tech/thrusts/tp322_attack_detec.html.

[8] The Boeing Company. *Neighborhood Key Information Distribution (NKID) Protocol (Draft)*, Boeing Document Number D658-10818-1, February 1998.

[9] Trusted Information Systems, Inc. *Intruder Detection Isolation Protocol (IDIP) Authentication Header (AH)*, TIS Report Number 0699D, November 1997.

[10] Trusted Information Systems, Inc., *IDIP AH with Hashed Message Authentication Codes (HMAC)-SHA-1,* TIS Report Number 0700D, November 1997.

[11] Trusted Information Systems, Inc., *Intruder Detection Isolation Protocol (IDIP) Encapsulating Security Payload (ESP)*, TIS Report Number 0698D, November 1997.

[12] Trusted Information Systems, Inc., *IDIP ESP with SKIPJACK Cipher Block Chaining (CBC),* TIS Report Number 0701D, November 1997.

[13] OpenSSL Project, http://www.openssl.org/.

[14] D. Schnackenberg, K. Djahandari, and D. Sterne, "Infrastructure for Intrusion Detection and Response", Proceedings of the DARPA Information Survivability Conference and Exposition, Hilton Head, SC, January 2000.

[15] LBNL's Network Research Group, "libpcap, the Packet Capture library", http://ee.lbl.gov/.

[16] Clifford Kahn, Don Bolinger, Dan Schnackenberg, "Communication in the Common Intrusion Detection Framework, v 0.7", June 1998.

[17] Secure Computing Corporation, Gauntlet Firewall, http://www.securecomputing.com/index.cfm?sKey=979.

[18] Secure Computing Corporation, Sidewinder, http://www.securecomputing.com/index.cfm?skey=232.

[19] Linux, http://www.linux.org/.

[20] IP Filter, http://coombs.anu.edu.au/~avalon/ip-filter.html.

[21] J. Epstein, "Architecture and Concepts of the ARGuE Guard", to be published in Proceedings of the 15th Annual Computer Security Applications Conference, Phoenix AZ, December 1999.

[22] G. Lamperillo, "Architecture and Concepts of the MPOG", NAI Labs Reference Number #0768, June 1999.

[23] NAI Labs, Firewall Toolkit. http://www.nai.com/research/nailabs/network-security/extended-secure.asp.

[24] TCP Wrappers http://cs-www.ncsl.nist.gov/tools/tools.htm.

[25] T. Fraser, L. Badger, M. Feldman, "Generic Software Wrappers "Hardening COTS Software with Generic Software Wrappers", Proceedings of the 1999 IEEE Symposium on Security and Privacy, IEEE, Oakland, California, May 1999.

[26] Net Squared, Network Radar, http://www.NetSQ.com/.

[27] Ulf Lindqvist and Phillip A. Porras, "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)", Proceedings of the 1999 IEEE Symposium on Security and Privacy, IEEE, Oakland CA, May 1999.

[28] C. Cowan, C. Pu, D. Maier, H. Hinton, P. Bakke, S. Beattie, A. Grier, P. Wagle, and Q. Zhang, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks", Proceedings of the 7th USENIX Security Conference, San Antonio TX, January 1998.

[29] M. Stillerman, C. Marceau, and M. Stillman, "Intrusion Detection for Distributed Applications", in Communications of the ACM 42:7, 1999.

[30] Network Associates, McAfee VirusScan, http://www.mcafee-at-home.com/products/virusscan/default.asp.

[31] Internet Security Systems, RealSecure, http://www.iss.net/products_services/enterprise_protection/rsnetwork/index.php.

[32] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle, "GrIDS -- A Graph-Based Intrusion Detection System for Large Networks", Proceedings of the 19th National Information Systems Security Conference, October 1996.

[33] L. Perrochon, W. Mann, S. Kasriel, and D. C. Luckham, "Event Mining with Event Processing Networks", Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining. Beijing, China, April 1999.

[34] Mountain Wave, Inc., "Adaptive Network Security Management", http://www.mountainwave.com/.

[35] Network Associates, Sniffer Technologies, http://www.sniffer.com/products/default.asp.

[36] "Adaptive System Security Policies Final Report", Boeing Document D950-10458-1. December 2000.

[37] D. Sterne, B. Wilson, K. Djahandari, D. Schnackenberg, H. Holliday, T. Reid, "Autonomic Response to Distributed Denial of Service Attacks", Proceedings of RAID 2001, Davis CA, October 2001.

6. GLOSSARY

AH	**AUTHENTICATION HEADER**
AIDE	Automated Intrusion Detection Environment
API	application programmer's interface
BSM	Basic Security Module
CBC	cipher block chaining
CEP	Complex Event Processing
CIDF	Common Intrusion Detection Framework
CISL	Common Intrusion Specification Language
CORBA	Common Object Request Broker Architecture
COTS	commercial off the shelf
DARPA	Defense Advanced Research Projects Agency
DES	Data Encryption Standard
DSA	Digital Signature Algorithm
DDoS	distributed denial of service
EMERALD	Event Monitoring Enabling Responses to Anomalous Live Disturbances
ESP	encapsulating security protocol
FTP	File Transfer Protocol
GrIDS	Graphical Intrusion Detection System
GUI	graphical user interface
HMAC	hashed message authentication code
HTML	Hyper-Text Markup Language
HTTP	Hyper-Text Transfer Protocol
IDIP	Intruder Detection and Isolation Protocol
IDMEF	Intrusion Detection Message Exchange Format
IETF	Internet Engineering Task Force
IFD	Integrated Feasibility Demonstration
IFE	Integrated Feasibility Experiment

IP	Internet Protocol
KEA	Key Exchange Algorithm
MPOG	Multi-Protocol Object Gateway
NKID	neighborhood key information distribution
SHA	Secure Hash Algorithm
SMARTS	System Management and Administration for Remote Trusted System
SNMP	Simple Network Management Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol